METHOD AND APPARATUS FOR RECORDING TRACE DATA IN A MICROPROCESSOR BASED INTEGRATED CIRCUIT

TECHNICAL FIELD OF THE INVENTION

The present invention relates in general to integrated circuit operation and more particularly to a method and apparatus for recording trace data in a microprocessor based integrated circuit.

BACKGROUND OF THE INVENTION

The ultimate test for the design of a microprocessor based integrated circuit is its operation in a system environment. However, the system environment provides little, if any, information about the internal state of the microprocessor to assist in diagnosing any failure that may occur during system operation. At best, external logic analyzers collect trace data from the system bus and secondary cache interface external to the microprocessor. More often, only the system trace data is captured as the secondary cache interface trace data is difficult mechanically and electrically to capture due to the complex network of short high frequency paths. Even if captured, these external signals provide no ability to determine the internal operation of the microprocessor. One key to solving this problem is to replicate the failure using a diagnostic program short enough to run in a chip tester and a simulator. The difficulty lies in the fact that the diagnostic program must accurately duplicate the processor state resulting in the failure. During actual operation of the microprocessor, its dynamic state greatly depends on branch predictions and cache refills. Out of order execution adds another level of complexity to any debugging efforts. Without guessing, this information is difficult to reconstruct.

Previous approaches to solving this problem include identifying what instruction was being executed upon the occurrence of a failure, tag an instruction and see how it executes, and counting events over an interval of time. These approaches do not provide information with respect to immediately preceding instructions which

usually are the initiating causes of a failure nor do they illustrate penalties for individual mispredicted branches or cache misses. Therefore, it is desirable to identify the internal state of a processor in order to identify causes of failure.

SUMMARY OF THE INVENTION

From the foregoing, it may be appreciated by those skilled in the art that a need has arisen to record information related to internal operation of a microprocessor in order to identify failures that occur during operation for appropriate correction. In accordance with the present invention, a method and apparatus for recording trace data in a microprocessor based integrated circuit are provided that substantially eliminate or greatly reduce disadvantages and problems of conventional system debugging techniques.

According to an embodiment of the present invention, there is provided an apparatus for recording trace data in a microprocessor based integrated circuit that includes an integrated circuit device with a central processing unit, an instruction cache, a data cache, and a trace recorder. During operation, the central processing unit provides trace information to the instruction cache and the data cache. The trace recorder is operable to selectively record the trace information in response to various trigger events in order to capture operational information that occurs around a trigger event. In this manner, failures in the operation of the central processing unit, the instruction cache, and the data cache may be corrected upon analyzing captured information associated with the failure.

The present invention provides various technical advantages over conventional system debugging techniques. For example, one technical advantage is to place a trace recorder on the integrated circuit with the microprocessor. Another technical advantage is to record information pertaining to a failure event during actual

operation that cannot be deduced from the operating program and external test equipment. yet another technical advantage is to use triggers to determine when and what to capture. Other technical advantages may be readily ascertained by those skilled in the art from the following figures, description, and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying drawings, wherein like reference numbers represent like parts, in which:

FIGURE 1 illustrates a block diagram of a microprocessor based integrated circuit;

FIGURE 2 illustrates a block diagram of a trace recorder of the microprocessor based integrated circuit;

FIGURE 3 illustrates a flow diagram of accessing configuration registers in the trace recorder in relation to its operating modes;

FIGURE 4 illustrates an example of logic for inhibiting the recording of data in the trace recorder;

FIGURE 5 illustrates a simplified block diagram of control logic for the trace recorder;

FIGURE 6 illustrates an example logic design for input logic of the control logic;

FIGURE 7 illustrates an example logic design for a trigger generator of the control logic;

FIGURE 8 illustrates an example logic design for a low address generator of the control logic;

FIGURE 9 illustrates an example logic design for a high address generator of the control logic;

FIGURE 10 illustrates a flowchart summarizing operation of the high address generator.

DETAILED DESCRIPTION OF THE INVENTION

FIGURE 1 is a block diagram of a microprocessor based integrated circuit 10. Integrated circuit 10 includes a central processing unit 12, an instruction cache 14, a data cache 16, a secondary cache 17, and a system interface 18. Integrated circuit 10 also includes a trace recorder 20 that has trigger logic 22, control logic 24, and a memory array 26. Trace recorder 20 captures and stores internal signals within integrated circuit 10 in its memory array 26 as determined by trigger logic 22 and control logic 24. A logic analyzer 28 or other type of test equipment may analyze the operation of integrated circuit 10 as it interfaces with an external cache 30 or other system elements through a system bus 32. Logic analyzer 28 may also be used to check the internal operation of integrated circuit 10 by analyzing internal signals captured in memory array 26 and provided through system interface 18.

Trace recorder 20 may operate in at least two modes, a read/write mode and a capture mode. In read/write mode, data may be read from or written to memory array 26. The read/write mode is initiated by a pair of command and data instructions, either MTC0/MFC0 instructions (read) or MTC0/MTC0 instructions (write). The first MTC0 instruction determines what element within trace recorder 20 is read or written to. After a read or a write occurs, trace recorder 20 is returned to a reset state. In capture mode, data is stored in memory array 26 as determined by trigger logic 22 and control logic 24. A single MTC0 instruction may be used to initiate the capture mode. Entries are written in each processor clock according to key signal values gathered from across

integrated circuit 10. Signals may be staged by two cycles before writing to prevent timing problems. Capturing continues until another MTC0 instruction disables capturing or a triggering event occurs.

In capture mode, memory array 26 records important signals. Since the size of memory array 26 is limited for incorporation onto an integrated circuit with a microprocessor, recording needs to be very selective. Detection of a special event as a trigger point is performed in order to mark the cycles. Examples of triggering events include CPU hung, memory addressing reaches a pre-determined address, and a register matches a pre-determined value. These trigger events are designed to lead to more clues about a specific bug or failure. The easiest method is to start recording data as soon as the triggering event occurs. However, more important information just prior to a triggering event may lead to determining a cause of the failure. Thus, information associated with a triggering event is captured and maintained prior to and subsequent to the occurrence of the triggering event. Captured information may be used to determine appropriate triggering events. Table 1 shows an example of the data format for captured information in memory array 26.

Table 1 Trace Recorder Cache Memory (TRCache) Data Format

Bit	# of Bit	Input Signal Source	Mux Select (Select Source 2)	Input Signal Source 2	Description
0	1	Inactive			see description for bit 15:8
1	1	Trigger			there was a trigger during the cycle
7:2	6	IF0D0IVA[7..2]			Instruction virtual address
15:8	8	IF0D0IVA[15..8]	TRCache[0]	InactiveCount	data hasn't changed for InactiveCount
16	1	IF0D0IVA[16]	IVASel	AQ-LinkBitN	ll/sc link bit
20:17	4	IF0D0IVA[20:17]	IVASel	PD0DCmd	CCBlk to AQ command or response
21	1	IF0D0IVA[21]	IVASel	NP0Store	AQ request was store
22	1	IF0D0IVA[22]	IVASel	(PD0ICmd !=0)	CCBlk to IFetch command or response
26:23	4	IF0D0IVA[26:23]	IVASel	DT???	LdSt address and Bank and Way info
27	1	IF0D0IVA[28]	IVASel	CD0WinnerNoneF	More MHT info
28	1	IF0D0IVA[28]	IVASel	SCDWrB	Data is being written to scache
31:29	3	IF0D0IVA[31:29]			

32	1	IFValidNotDecode			any instructions valid but not decoded
38:33	6	GR0D0ActQTag0[5:0]			Active list write pointer
44:39	6	GRactctl.0.RdPtr[5..0]			Active list read pointer
45	1	GR0InExc			"Interrupt" type of exception
46	1	GR2W0ExcPendB			Other type of exception
47	1	DT2E2LoadDone			LoadDone
48	1	NP0IFGoes			IFetch request sent to MHT
49	1	NP0LSGoes			AQ request to MHT
54:50	5	CDValid[4..0]			Valid entries in MHT
55	1	PC0PrcReqRdy			
56	1	PR9SysGntInB	SysTrVal	SysCmd[4]	
57	1	PR9SysValInB	SysTrVal	SysCmd[5]	
58	1	EA0SysValOutB	SysTrVal	SysCmd[6]	
59	1	PR9SysRespValInB	SysTrVal	SysCmd[7]	
61:10	2	PR9SysRespIn[1:0]	SysTrVal	SysCmd[9:8]	
62	1	SysCmd[11]	SysTrVal	SysCmd[10]	
63	1	SysTrVal			Set when the Source 1 group of traced System interface sigs are valid. Source 2 group is valid in next cycle unless inactive indicator is set.

FIGURE 2 is a block diagram of trace recorder 20. The main functional components of trace recorder 20 include memory array 26, control logic 24, and trigger logic 22. Trigger logic 22 uses configuration registers to implement the capture and trigger technique for trace recorder 20. These registers include a trigger control register 30, a capture control register 32, an order map register 34, a trigger address register 36, and inhibit mask registers 38. These registers set up the signal capture so that the most important segment of the signal traces are written into memory array 26. FIGURE 3 shows a flow diagram of accessing the configuration registers in relation to the read/write and capture modes discussed above.

Trigger control register 30 provides enable and address signals for trace recorder 20. These signals are shown in Table 2. Trigger control register 30 generates a capture array index signal, a memory select signal, a global enable signal, and a capture indicator signal. The capture array index signal provides the addresses to memory array 26 to perform reads and writes in the

read/write mode. In the capture mode, this signal provides the current recording pointer for profiling. The global enable signal provides the main enabling power for the other configuration registers and memory array 26 in trace recorder 20. The capture indicator signal provides a toggle indication as to whether or not data is to be captured. A single MFC0 instruction prior to a MTC0 instruction allows for reading of trigger control register 30.

Table 2 Trigger Control Register

Bit	Field Name	Description
8:0	CAIdx	RW Mode: Index for the 512 entry capture memory In Capture Mode: Current recording index pointer After Capture Mode: Stop pointer
11:9	reserved	
15:12	MemSel	Memory Element Selection and Status MemSel =0: RW Command Mode; Select Trade Recorder Control Register. MemSel!=0: RW Data Mode; Select MemSel=1,2,3,5,6,7,12,13,14
16	GEnable	Global Enable Power Up
17	CIBit	Capture Indicator

The memory select signal determines which configuration register of trace recorder 20 is selected or which portion of memory array 26 is desired. Table 3 shows the breakdown of the memory select signal. Memory array 26 and the configuration registers are directly writable to test and load the memory elements and directly readable to read data. Reading and writing is performed by executing a MTC0 instruction that sets the memory select signal. Another MTC0 or MFC0 instruction provides the data to be written or read out and, upon execution, clears the memory select signal. The default value for the memory select signal is zero. With the memory select signal at zero, trace recorder 20 is in a command mode waiting for a command MTC0 instruction in order to prepare the appropriate setup. When the memory

select signal is not zero, trace recorder 20 awaits for a data MTC0 or data MFC0 to complete the write or read function. After completion, the memory select signal is returned to the zero state.

Table 3 Memory Selection

MemSel	# of Bits	Description	Condition
0	18	Select the Trace Recorder Control Register	MTC0, MFC0
1	32	Select Capture Control Register	MTC0, MFC0
2	32	Select Order Map and Status Register	MTC0, MFC0
3	32	Select Trigger Address Register	MTC0, MFC0
5	32	Select bit 31:0 of the Capture Array	MTC0, MFC0, CI=0
6	32	Select bit 63:32 of the Capture Array	MTC0, MFC0, CI=0
7	8	Select bit 71:64 of the Capture Array	MTC0, MFC0, CI=0
12	32	Select Recording Inhibit Mask Register 0	MTC0, MFC0
13	32	Select Recording Inhibit Mask Register 1	MTC0, MFC0
14	32	Select Recording Inhibit Mask Register 2	MTC0, MFC0

Capture control register 32 specifies how the capture is to occur and controls maintaining the data once it is captured. Table 4 shows what may be included in capture control register 32.

Table 4 Capture Control Register

Bit	Field Name	Description
0:11	NCycleTrigger	TEvCPUHung=1: NCycleTrigger defines the number of cycles that CPU hangs. TEvCPUHung=0: A trigger is generated for every NCycleTrigger cycles.
14:12	OldestPre	The oldest block number for current Pre-Trigger Buffer
22:15	MaxCount	NthCycleMode=1: Inhibit MaxCount-1 cycles/events. Profiling NEventMode : Record MaxCount events. All other modes: MaxCount=255.
23	LastTMode	LastTMode=1: Last Trigger is recorded. LastTMode=0: First Trigger is recorded.
24	NEventMode	Interval Profiling, record MaxCount events where MaxCount<64
25	NThCycleMode	Nth cycle sampling mode, record the Nth cycle where N=MaxCount<256.
26	TEvIVAMatch	Enable trigger event of IVA match
27	TEvWatchR	Enable trigger event of read access to address in Watch
28	TEvCPUHung	Enable CPU hung trigger event
29	TEvNCycles	Enable trigger generated every N cycles. TEvCPUHung must be zero.
30	reserved	
31	EverTriggered	Whether the trigger ever happened

The NCycleTrigger signal determines how a trigger signal is generated. A trigger may be generated for every NCycleTrigger cycles when the TEvNCycles signal is enabled and the TEvCPUHung signal is disabled. With both

the TEvNCycles and TEvCPUHung signals disabled, a trigger may be generated for every NCycleTrigger processor cycles. With the TevCPUHung signal enabled, the NCyclesTrigger signal defines the cycles that the CPU hangs. A CPU hung trigger is preferably based on a free running 12 bit processor clock counter that is reset whenever an instruction graduates. When the counter overflows, the CPU hung trigger is asserted. This allows for the capturing of activity leading up to a processor hang since after the hang the CPU may still be responding to interventions.

The OldestPre signal indicates the oldest valid location in memory array 26 prior to receipt of a trigger. The MaxCount signal provides for the recording of data for the number of events specified when the NEventMode signal is enabled. The MaxCount signal also provides for the recording of data for Nth cycle sampling upon enablement of the NTHCycleMode signal. A TEvIVAMatch signal, upon being enabled, causes a comparison of the contents of trigger address register 34 to bits in an IVA address. Upon a match and determination of a valid decoded instruction, a trigger may be generated. When a TEvWatchR signal is enabled, a trigger may be generated if either a read or write data access is made to the physical memory address in a CPU watch register. An EverTriggered signal informs as to the occurrence of at least one trigger, indicating whether useful data has been captured in memory array 26.

There are at least three types of recording modes that determine how to start and stop capturing data around a triggering event. These recording modes include a last trigger, a first trigger, and profiling. The

LastTMode signal determines which of the last trigger and first trigger recording modes are implemented. For last trigger enablement, the data around the last trigger is recorded and maintained in memory array 26. For first trigger enablement, recording stops a desired number of cycles after the occurrence of the first trigger and the data is maintained in memory array 26 despite the occurrence of other triggers. Last trigger and first trigger enablement may also be implemented only for every Nth cycle or Nth event. The other type of recording mode is profiling wherein a number of events after a trigger are recorded. In profiling mode, there is at least one trigger every specified number of events. Table 5 summarizes the preferable recording modes.

Table 5 Recording Modes

Name	NthCycleMod	NEventMode	LastTMode	TExCPUHung	Stop Method	Data Format	Recording Inhibit	Limit
Last Trigger	0	0	1	X	MTCO Reset	Order Map	data no change	NPre=1,...,4;NPost =0,1,...,4
First Trigger	0	0	0	X	NPost met resets CI	Order Map	data no change	NPre=1,...,7;NPost =0,1,...,7
Profiling	0	1	1	0	MTCO Reset	Continue	data no change	NEvents <64
Profiling	0	1	0	0	RAM is Full, Reset CI	Continue	data no change	NEvents <64
Nth Cycle	1	0	1	X	MTCO Reset	Order Map	Count<N	N<256
Nth Cycle	1	0	0	X	NPost met resets CI	Order Map	Count<N	N<256
Nth Event	1	1	1	X	MTCO Reset	Order Map	Count<N	N<256
Nth Event	1	1	0	X	NPost met resets CI	Order Map	Count<N	N<256

Order map register 34 specifies the ordering for data as it is recorded in memory array 26. Table 6 shows what may be included in order map register 34. Preferably, memory array 26 is partitioned into 8 blocks with each block being available in a desired order to record data. The OrderMap signals provide an address for one of the 8 blocks and establishes the ordering of the recorded data. The NPre signal specifies the number of

blocks for recording and keeping before an occurrence of a trigger. The NPost signal specifies the number of blocks for recording and maintaining after the occurrence of a trigger. The ShiftWrap signal indicates a Pre-Trigger wrap-around state. The StatePost signal indicates a Post-Trigger state. If the ShiftWrap and StatePost signals are disabled, then ordering is in a PreNoWrap state with no rearranging ordering. If the ShiftWrap signal is enabled, then ordering is in a WaitTrigger state and wrap around shift reordering is implemented. If the StatePost signal is enabled, then ordering is in a post-Trigger state with no rearranging ordering. Preferably, the initial state is PreWrapNo unless the NPre signal is zero wherein the initial state is WaitTrigger. Further information on a specific order map implementation can be found in copending U.S. Application Serial No. _____ entitled "Device and Method for Storing Information in Memory" which is hereby incorporated by reference herein.

Table 6 Order Map and Status Register

Bit	Field Name	Description
2:0	OrderMap0	Order Map values at entry 0
5:3	OrderMap1	Order Map values at entry 1
8:6	OrderMap2	Order Map values at entry 2
11:9	OrderMap3	Order Map values at entry 3
14:12	OrderMap4	Order Map values at entry 4
17:15	OrderMap5	Order Map values at entry 5
20:18	OrderMap6	Order Map values at entry 6
23:21	OrderMap7	Order Map values at entry 7
26:24	NPre	The number of blocks in Pre-Trigger buffer.
29:27	NPost	The number of blocks in Post-Trigger buffer.
30	ShiftWrap	The state variable indicating the Pre-Trigger wrap-around state
31	StatePost	The state variable indicating the Post-Trigger state

FIGURE 4 shows example logic for inhibiting the recording of data in memory array 26. Inhibit mask registers 38 provide a capability to inhibit the

recording of data. To make efficient use of the limited memory space within memory array 26, cycles are recorded only when specific criteria is met and other cycles are skipped. When the capture indicator signal of trigger control register 30 is enabled, memory array 26 will capture activity every cycle if it is not inhibited. There are at least four inhibit signals with appropriate masks that perform the inhibit operation. Table 7 shows these recording inhibit signals.

Table 7 Recording Inhibit Signals

Inhibit Signal Name	Mask Name	Description
NoChangePClk	Signal Mask Register	No change for signals synchronized with processor clock.
NoChangeSysClk	SysAD Inhibit Mask	No change for signals synchronized with SysClk.
KerUsrExc	KerUsrExc Inhibit Mask	Wether program is in Kernal/User and/or Exception mode.
Count<N	NCycle	Skip N cycles

The NoChangePClk signal detects for changes of certain signals when synchronized with the processor clock through masking with first and second inhibit masks. If there is no change in the data, then data is not recorded. Tables 8, 9, and 10 show examples of inhibit mask registers that may be used.

Table 8 Recording Inhibit Mask Register 0

Mask Bit	Name of Signals Masked	# Bit	Description
22:0	rserved	23	reserved
23	Ivasel	1	IVASEL = 0 select ld/st MHT degub signals IVASEL = 1 select IFODOIVA [31:16]
27:24	KerUrsExc Inhibit Mask	4	bit 27: Inhibit when process is in exception and user mode. bit 26: Inhibit when process is in exception mode, but not in user mode. bit 25: Inhibit when process is not in exception mode, but in user mode. Bit 24: Inhibit when process is not in exception mode, not in user mode.
31:28	SysAD Inhibit Mask	4	bit 27: Inhibit when PR9SysRespValInB is asserted. bit 27: Inhibit when PR9SysValInB is asserted and SysCmd[11]=0. bit 27: Inhibit when PR9SysValInB is asserted and SysCmd[11]=1. bit 27: Inhibit when PR9SysGntB changes.

Table 9 Recording Inhibit Mask Register 1

Mask Bit	Name of Signals Masked	# Bit	Description
0	reserved	1	reserved
31:1		31	mask signals going to field 31:1 of Trace Recorder Cache Memory

Table 10 Recording Inhibit Mask Register 2

Mask Bit	Name of Signals Masked	# Bit	Description
22:0		23	mask signals going to field 54:32 of Trace Recorder Cache Memory
31:23	reserved	9	reserved

The KerUsrExc signal indicates whether the program is in a user and/or exception mode. Inhibit may occur if either, neither, or both modes are asserted. This inhibit may be used in conjunction with certain bits of the processor status register. The CZ0KSUXD signal indicates that CPU 12 is in user mode and CZ0EXLXorERLX indicates that CPU 12 is in exception mode. Table 11 shows when the KerUsrExc signal is asserted.

Table 11 KerUsrExc Inhibit

CZ0KSUXD	CZ0EXLXorERLX	KerUsrExc Inhibit Signal
0	0	1 if bit[24]=1
0	1	1 if bit[25]=1
1	0	1 if bit[26]=1
1	1	1 if bit[27]=1

The NoChangeSysClk signal detects for changes in the SysAD signals synchronized with the system clock. No change in data will result in no data being recorded. Table 12 shows when the NoChangeSysClk signal is asserted.

Table 12 SysAD Inhibit

	SysCmd[11]	NoChangeSysClk	Comments
PR9SysGntB changes	X	1 if bit[28]=1	
PR9SysValInB asserted	1	1 if bit[29]=1	valid SysAD data
PR9SysValInB asserted	0	1 if bit[30]=1	valid SysAD command
PR9SysRespValInB asserted	X	1 if bit[31]=1	

The Count<N signal provides for capturing of data every Nth cycle and inhibits for the intervening N-1 cycles. A trigger cycle is preferably recorded despite a Count<N inhibit request. Table 13 shows a summary of when recording is performed or inhibited.

Table 13 Recording Inhibit

Action	KerUsrExc Inhibit	NThCycle Mode	Count<N Inhibit	NoChangeAll Inhibit	
Record	0	X	0	0	
Record	0	0	X	0	
Inhibit	0	0	X	1	
Record	0	1	0	X	
Inhibit	0	1	1	X	
Inhibit	1	X	X	X	

FIGURE 5 shows a block diagram of control logic 24. The function of control logic 24 is to generate the memory addresses and write enables to memory array 26 for capture mode operation and to update the configuration registers. Control logic 24 includes input logic 40, a trigger generator 42, a low address generator 44, and a high address generator 46. FIGURE 6 shows an example logic design for input logic 40. Input logic 40 detects input signal changes and generates an inactivate count. FIGURE 7 shows an example logic design for trigger generator 42. Trigger generator 42 generates a trigger signal corresponding to a triggering event. Low address generator 44 generates the lower address field in accessing memory array 26. It also updates the trigger index and the ever triggered status bit. An example of logic for low address generator 44 is shown in FIGURE 8. The high address generator 46 generates the higher address field in accessing memory array 26. An example of logic for high address generator 46 is shown in FIGURE

9. A flowchart summarizing the operation of high address generator 46 is shown in FIGURE 10.

Thus, it is apparent that there has been provided, in accordance with the present invention, a method and apparatus for recording trace data in a microprocessor based integrated circuit that satisfies the advantages set forth above. Although the present invention has been described in detail, it should be understood that various changes, substitutions, and alterations may be readily ascertainable by those skilled in the art and may be made herein without departing from the spirit and scope of the present invention as defined by the following claims.